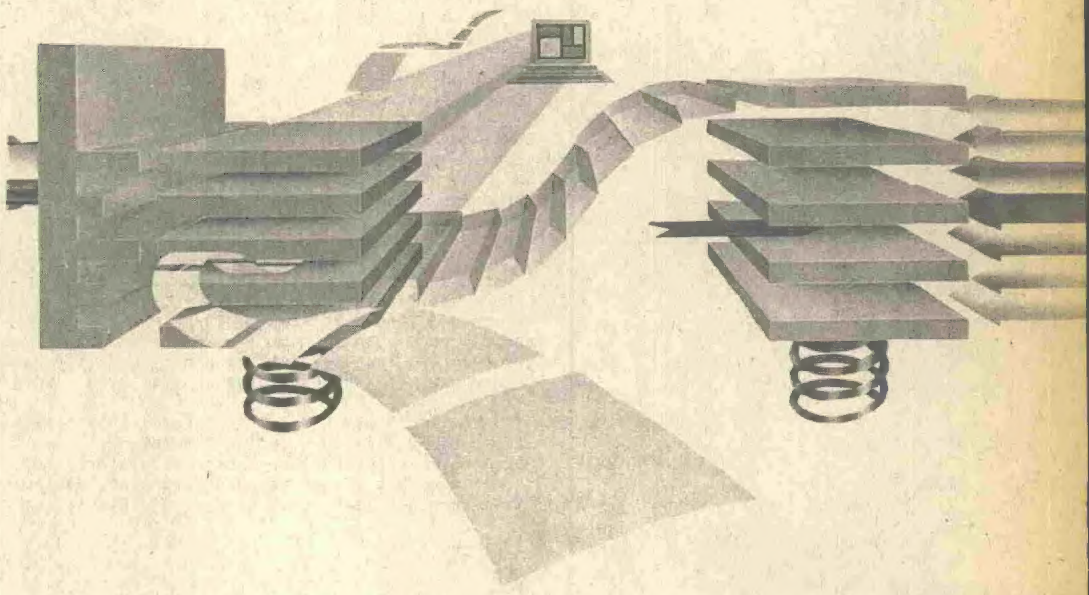
ซอฟต์แวร์
สมบัติ เตรียมแจ้งอรุณ

OFT
AIL

การนำหน่วยความจำเพิ่มขยายมาใช้

*คนิคการใช้ประโยชน์จาก
น่วยความจำ EMS
ยใช้ภาษาปาสคาล*



นปัจจุบัน หน่วยความจำ 640 กิโลไบต์ ของ IBM PC ดูจะยเกินไป เนื่องมาจากการพัฒนาฟต์แวร์เป็นไปอย่างรวดเร็ว เช่น mmand shells, LAN shells, device vers และ Resident programs ล้วนงใช้พื้นที่หน่วยความจำทั้งสิ้น เป็นให้หน่วยความจำที่เหลือใช้น้อยลงอีก การใช้หน่วยความจำเพิ่มขยาย xpanded Memory (EMS)) เป็นทางแก้ปัญหาอันหนึ่ง ซึ่งมีแนวโน้มนำมาใช้กันมากขึ้น ในฉบับนี้ผู้เขียนเสนอวิธีการนำหน่วยความจำเพิ่มายมาใช้ในโปรแกรมของเรา ซึ่งต่อผู้เขียนขอเรียกหน่วยความจำเพิ่มายว่า EMS

EMS สามารถใช้ในไมโครคอมเตอร์ที่ใช้ CPU เบอร์ 8086, 8088, 286 และ 80386 ซึ่งต่างจากหน่วยามจำยืดขยาย (Extended Memory) ไม่สามารถใช้กับ CPU เบอร์ 8086 ะ 8088 ได้ ก่อนจะใช้ EMS จำเป็นงมีการ์ดเพิ่มหน่วยความจำ และฟต์แวร์ซึ่งสนับสนุนการใช้ EMS xpanded Memory Manager Driver MM))

การติดต่อกับ EMS จะกระทำยผ่าน Interrupt หมายเลข 67h โดยปรกติจะใช้ Registers ต่าง ๆ ดังนี้

- AH เป็นหมายเลขของ Function
- AL เป็นหมายเลขของ Subfunction (ถ้ามี)
- BX เป็นหมายเลขของ Pages
- DX เป็นหมายเลขของ Handle ซึ่ง EMM ใช้ในการเลือก Pages

หลักการทำงานของ EMS เป็น Bank Switching (ผู้อ่านสามารถดูรายละเอียดได้จากไมโครฯ 64) ในพื้นที่หน่วยความจำ 64 กิโลไบต์ (Page Frame) ซึ่งอยู่เหนือหน่วยความจำปรกติ 640 กิโลไบต์ใน EMM จะใช้ Pages 2 ชนิดคือ Physical Page และ Logical Page Physical Page จะอยู่ใน Page Frame ซึ่งตามปรกติจะประกอบด้วย 4 Physical Pages (0-3) โดยแต่ละ Physical Page จะมีขนาด 16 กิโลไบต์ ส่วน Logical Page จะอยู่ในการ์ด EMS ซึ่งไม่สามารถเรียกได้โดยตรง จำเป็นต้องเรียกผ่าน EMM

การใช้ EMS มีขั้นตอนดังต่อไปนี้ (ดูโปรแกรมในรูปที่ 1 ประกอบ)

1. ตรวจสอบว่า EMM ได้รับการติดตั้งแล้วหรือไม่ โดยตรวจสอบที่ Offset 10 ของค่าเซกเมนต์ที่อินเตอร์รัพต์เวคเตอร์หมายเลข 67h ชี้ หากพบข้อความว่า 'EMMXXXX0' ก็แสดงว่า EMM ได้รับการติดตั้งแล้ว

2. ตรวจสอบการให้บริการของ EMM โดยเรียก Procedure GetStatus

3. ตรวจสอบว่ามีหน่วยความจำ EMS เพียงพอกับความต้องการหรือไม่ โดยเรียก Procedure GetNumberOfPages

รูปที่ 1 โปรแกรมยูนิต EMS32 รวบรวมบริการของ EMS

```
Unit EMS32;
Interface
uses DOS;
Type
    HandleRec = Record
                  Handle,Pages : Word;
                end;
    HandleAr  = Array[1..256] of HandleRec;
Var
    EmmInstalled : Boolean;
    EmmMaxAvail,
    EmmTotal     : Word;
    EmmSeg       : Word;
    EmmVer       : Byte;

Function EmmError : Boolean;
Procedure ReportError;

(*************LIM EMS 3.0 or later***********)
Procedure GetStatus;
Procedure GetPageframe (Var pageSeg : Word);
Procedure GetNumberOfPages (Var total, unAlloc : Word);
Procedure AllocateHandleAndPages (pages : Word; Var handle : Word);
Procedure MapPage (physPage, logPage, handle : Word);
Procedure ReleaseHandle (handle : Word);
Procedure GetVersion (Var version : Byte);
Procedure SavePageMap_30 (handle : Word);
Procedure RestorePageMap_30 (handle : Word);
Procedure GetHandleCount (Var num : Word);
Procedure GetHandlePages (handle : Word; Var pages : Word);
Procedure GetPagesForAllHandles (Var size : Word; Var ar : HandleAr);

(*************LIM EMS 3.2 or later***********)
Procedure SavePageMap (point : Pointer);
Procedure RestorePageMap (point : Pointer);
Procedure Save_RestorePageMap (srcPoint, destPoint : Pointer);
Procedure GetSizePageMap (Var size : Byte);

Implementation

Var
  reg   : Registers;
  error : Byte;

Function EmmError : Boolean;
Begin
  EmmError := (error <> 0);
end;

Procedure ReportError;
Begin
  error := reg.AH;
  if (EmmError) Then
     Case (reg.AH) Of
       $80 : Writeln ('Internal EMM software error. Possible Memory Corruption');
       $81 : Writeln ('Problem in EMM hardware.');
       $82 : Writeln ('EMM busy.');
       $83 : Writeln ('Invalid handle.');
       $84 : Writeln ('No such function exists.');
       $85 : Writeln ('No more handles.');
       $86 : Writeln ('Error saving/restoring mapping context.');
       $87 : Writeln ('Physically not enough pages. No pages allocated.');
       $88 : Writeln ('Not enough available pages. No pages allocated.');
       $89 : Writeln ('Can not allocate 0 pages');
       $8A : Writeln ('Logical page is outside range of pages assigned to this handle.');
       $8B : Writeln ('Illegal physical page number (not [0..3]).');
       $8C : Writeln ('Page Mapping save area full.');
       $8D : Writeln ('Save area already contains context with this handle.');
       $8E : Writeln ('Save area does not contain context with this handle.');
       $8F : Writeln ('No such subfunction exists.');
     end;
end;

Procedure GetStatus;
Begin
  reg.AH := $40;
  Intr ($67, reg);
  ReportError;
end;

Procedure GetPageframe (Var pageSeg : Word);
Begin
```

→

รูปที่ 1 (ต่อ)

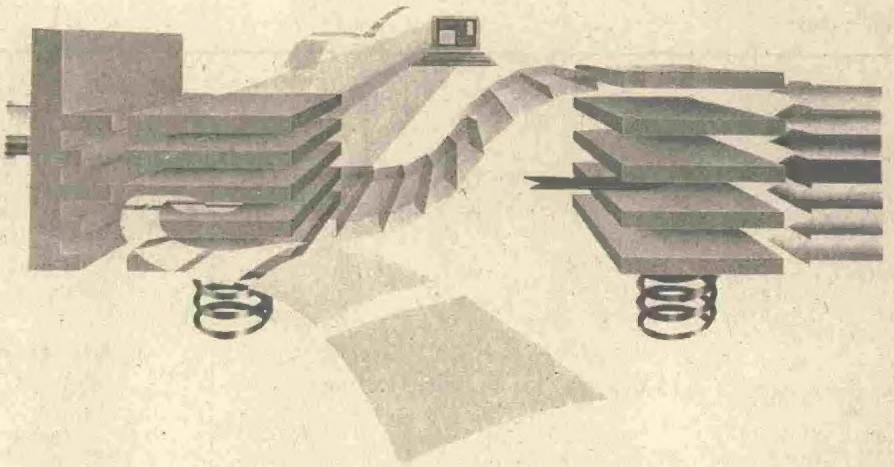
```
 reg.AH := $41;
 Intr ($67, reg);
 pageSeg := reg.BX;
 ReportError;
nd;

rocedure GetNumberOfPages (Var total, unAlloc : Word);
egin
 reg.AH := $42;
 Intr ($67, reg);
 unAlloc := reg.BX;
 total   := reg.DX;
 ReportError;
nd;

rocedure AllocateHandleAndPages (pages : Word; Var handle : Word);
egin
 reg.AH := $43;
 reg.BX := pages;
 Intr ($67, reg);
 handle := reg.DX;
 ReportError;
nd;

rocedure MapPage (physPage, logPage, handle : Word);
egin
 reg.AH := $44;
 reg.AL := physPage;
 reg.BX := logPage;
 reg.DX := handle;
 Intr ($67, reg);
 ReportError;
nd;

Procedure ReleaseHandle (handle : Word);
Begin
  reg.AH := $45;
  reg.DX := handle;
  Intr ($67, reg);
  ReportError;
end;

Procedure GetVersion (Var version : Byte);
Begin
  reg.AH := $46;
  Intr ($67, reg);
  version := reg.AL;
  ReportError;
end;

Procedure SavePageMap_30 (handle : Word);
Begin
  reg.AH := $47;
  reg.DX := handle;
  Intr ($67, reg);
  ReportError;
end;

Procedure RestorePageMap_30 (handle : Word);
Begin
  reg.AH := $48;
  reg.DX := handle;
  Intr ($67, reg);
  ReportError;
end;

Procedure GetHandleCount (Var num : Word);
Begin
  reg.AH := $4B;
  Intr ($67, reg);
  num := reg.BX;
  ReportError;
end;

Procedure GetHandlePages (handle : Word; Var pages : Word);
Begin
  reg.AH := $4C;
  reg.DX := handle;
  Intr ($67, reg);
  pages := reg.BX;
  ReportError;
end;

Procedure GetPagesForAllHandles (Var size : Word; Var ar : HandleAr);
Begin
  reg.AH := $4D;
  reg.ES := Seg (ar);
  reg.DI := Ofs (ar);
  Intr ($67, reg);
  size   := reg.BX;
  ReportError;
end;

Procedure SavePageMap (point : Pointer);
Begin
  reg.AH := $4E;
  reg.AL := $00;
  reg.ES := Seg (point^);
  reg.DI := Ofs (point^);
  Intr ($67, reg);
  ReportError;
end;

Procedure RestorePageMap (point : Pointer);
Begin
  reg.AH := $4E;
  reg.AL := $01;
  reg.DS := Seg (point^);
  reg.SI := Ofs (point^);
  Intr ($67, reg);
  ReportError;
end;

Procedure Save_RestorePageMap (srcPoint, destPoint : Pointer);
Begin
  reg.AH := $4E;
  reg.AL := $02;
  reg.DS := Seg (srcPoint^);
  reg.SI := Ofs (srcPoint^);
  reg.ES := Seg (destPoint^);
  reg.DI := Ofs (destPoint^);
  Intr ($67, reg);
  ReportError;
end;

Procedure GetSizePageMap (Var size : Byte);
Begin
  reg.AH := $4E;
  reg.AL := $03;
  Intr ($67, reg);
  size   := reg.AL;
  ReportError;
end;

Type
  arType = array[1..8] of Char;
Var
  p : ^arType;
Begin
  EmmInstalled := FALSE;
  GetIntVec ($67, Pointer (p));
  p := ptr (Seg (p^),$0A);
  If (p^ = 'EMMOOOOX') Then
  Begin
    GetStatus;
    If (Not EmmError) Then EmmInstalled := TRUE;

    GetNumberOfPages (EmmTotal, EmmMaxAvail);
    if (EmmError) then Halt;

    GetPageFrame (EmmSeg);
    if (EmmError) then Halt;

    GetVersion (EmmVer);
    if (EmmError) Then halt;
  end;
```

■

โปรแกรม EMS32.PAS ในรูปที่ 1 เป็น Unit ซึ่งรวบรวมการให้บริการของ EMM ซึ่งสรุปได้ดังนี้

Function EmmError และ Procedure ReportError

- รายงานข้อผิดพลาดที่เกิดขึ้น



สำหรับ LIM EMS 3.0 หรือมากกว่า

Procedure GetStatus

- ตรวจสอบการให้บริการของ EMM

Procedure GetPageframe

- หาค่าเซกเมนต์ของ Page Frame

Procedure GetNumberOfPages

- หาจำนวนของ Pages ทั้งหมดและที่ยังว่างอยู่ใน EMS

Procedure AllocateHandleAndPages

- จอง Logical Pages และหา Handle ใน EMS

Procedure MapPage

- แม็บ Logical Page ใน Handle กับ Physical Page ใน Page Frame

Procedure ReleaseHandle

- ถอนการจอง Logical Pages และ Handle ใน EMS

Procedure GetVersion

- ตรวจสอบเวอร์ชันของ EMM

Procedure SavePageMap_30

- เก็บค่าใน Registers ซึ่งจะเก็บตำแหน่งของ Handle

Procedure RestorePageMap_30

- เรียกค่าใน Registers คืนซึ่งเก็บโดย Procedure SavePageMap_30

Procedure GetHandleCount

- ตรวจสอบจำนวน Handle ที่แอคทีฟ

Procedure GetHandlePages

- ตรวจสอบจำนวน Logical Pages ของ Handle

Procedure GetPagesForAllHandles

- ตรวจสอบจำนวน Logical Pages และ Handle

สำหรับ LIM EMS 3.2 หรือมากกว่า

Procedure SavePageMap

- เหมือน Procedure SavePageMap_30 แต่ใช้พอยน์เตอร์ชี้ Buffer แทน

Procedure RestorePageMap

- เรียกค่าในรีซิสเตอร์คืนซึ่งเก็บโดย Procedure SavePageMap

Procedure Save_RestorePageMap

- เก็บค่า Handle เดิม และเรียก Handle ใหม่

Procedure GetSizePageMap

- ตรวจสอบขนาด Buffer ที่ใช้เก็บรายละเอียดของการจัด Pages

ตัวแปรใน Unit ที่สำคัญ สรุปได้ดังนี้

EmmInstalled

รูปที่ 2 โปรแกรมตัวอย่างแสดงการใช้ยูนิต EMS32

```
Program Example;
Uses CRT,EMS32,GRAPH;
Type
  HandleRec = Record
                H : Word;
                P : Pointer;
              end;
  HandleArr = Array[1..10] of HandleRec;
Var
  gDriver,gMode : Integer;
  loop1,loop2   : Integer;
  storage       : HandleArr;

Function LargestBlock : Word;
Var
  y : Word;
Begin
  y := 0;
  While (y < GetMaxY) do
  Begin
    if (Imagesize (0,0,GetMaxX,y) = $FFFF) Then
    Begin
      LargestBlock := (y-1);
      Exit;
    end;
    Inc(y);
  end;
  LargestBlock := y;
end;

Procedure SwapGraph (toMem : Boolean);
Var
  blocks, size : Word;

  Procedure SwapToEms;
  Begin
    AllocateHandleAndPages ((blocks*4),storage[1].H);
    If (EmmError) Then Halt;
    storage[1].P := Ptr(EmmSeg,0);
    For loop1 := 1 to blocks do
    Begin
      For loop2 := 0 to 3 do
      Begin
        MapPage (loop2, ((loop1-1)*4)+loop2, storage[1].H);
        If (EmmError) Then Halt;
      End;
      GetImage (0, (loop1-1)*size, GetMaxX, loop1*size, storage[1].P^);
    End;
  End;

  Procedure SwapFromEms;
  Begin
    storage[1].P := Ptr(EmmSeg,0);
    For loop1 := 1 to blocks do
    Begin
      For loop2 := 0 to 3 do
      Begin
        MapPage (loop2, ((loop1-1)*4)+loop2, storage[1].H);
        If (EmmError) Then Halt;
      End;
      PutImage (0, (loop1-1)*size, storage[1].P^, normalPut);
    End;
    ReleaseHandle (storage[1].H);
    If (EmmError) Then Halt;
  End;

  Procedure SwapToRam;
  Begin
    For loop1 := 1 to blocks do
    Begin
      GetMem(storage[loop1].P, $FFFF);
      storage[loop1].H := $FFFF;
      If (storage[loop1].P = NIL) Then Halt;
      GetImage (0, (loop1-1)*size, GetMaxX, loop1*size, storage[loop1].P^);
    End;
  End;

  Procedure SwapFromRam;
  Begin
```

→

- มีค่า TRUE เมื่อ EMM driver ได้รับการติดตั้งแล้ว

EmmMaxAvail

- เป็นจำนวน 16k Pages ที่เหลืออยู่

EmmTotal

- เป็นจำนวน 16k Pages สูงสุดที่มีในฮาร์ดแวร์

EmmSeg

- เป็นค่าเซกเมนต์ของ Page Frame

EmmVer

- เป็นเวอร์ชันของ EMM โดยสี่บิตบนเป็นเลขหลัก และสี่บิตล่างเป็นจุดทศนิยม เช่น EMS V.3.2. จะมีค่า EmmVer เท่ากับ $32

การติดต่อกับหน่วยความจำ EMS ทำได้โดยการแม็บ Logical Page ที่ต้องการกับ Physical Page จากนั้นก็เพียงติดต่อกับ Physical Page ไม่ว่าจะเขียนหรืออ่าน ก็เสมือนได้กระทำกับ Logical Page นั้น โดย EMM จะจัดการให้โดยอัตโนมัติ

ในที่นี้ จะขอยกโปรแกรมตัวอย่างในรูปที่ 2 โดยสมมุติว่า กำลังอยู่ในกราฟิกโหมด และต้องการกลับไปสู่เท็กซ์โหมด โดยจะเก็บจอภาพไว้ใน EMS และกลับมาอยู่ในกราฟิกโหมดโดยเรียกข้อมูลจอภาพออกมาจาก EMS โปรแกรมจะเข้าสู่กราฟิกโหมด อย่าลืมว่าจะต้องมีไฟล์ BGI ของจอภาพที่ใช้อยู่ โปรแกรมจะวาด Pattern บนจอภาพ และเก็บข้อมูลจอภาพไว้ใน EMS เมื่อกดแป้น <ENTER> โปรแกรมจะเข้าสู่เท็กซ์โหมดแสดงข้อมูลรายละเอียด

```
    For loop1 := 1 to blocks do
    Begin
      PutImage (0, (loop1-1)*size, storage[loop1].P^, normalPut);
      FreeMem(storage[loop1].P,$FFFF);
    End;
  End;

Begin
  size := LargestBlock;
  blocks := GetMaxY div size;
  If ((GetMaxY mod size) > 0) Then Inc(blocks);
  If (EmmInstalled) and (EmmMaxAvail >= (blocks*4)) Then
     If (toMem) Then
        SwapToEms
     Else
        SwapFromEms
  Else
     If (toMem) Then
        SwapToRam
     Else
        SwapFromRam;
End;

Procedure DisplayEmmInfo;
Var
  tmpTotal, tmpUnAlloc : Word;
Begin
  GetNumberOfPages (tmpTotal,tmpUnAlloc);
  Writeln('The graphic image has been swapped to memory.');
  If (EmmInstalled) Then
     Writeln('There are ',tmpUnAlloc,' pages of EMS left.');
  If (storage[1].H <> $FFFF) Then
  Begin
    GetHandlePages (storage[1].H, tmpTotal);
    Write('There were ', tmpTotal,' pages allocated to');
    Writeln('store the screen with.');
  End;
End;

Begin
  Clrscr;
  If (EmmInstalled) Then
  Begin
    Writeln( LongInt(EmmTotal)*16,'k of EMS total on this system.');
    Writeln( EmmMaxAvail,' pages available. (',LongInt(EmmMaxAvail)*16,'k).');
    Readln;
  End;
  gDriver := Detect;
  InitGraph(gDriver, gMode, '');
  If (GraphResult <> grOk) Then Halt;
  if GetMaxColor > 1 Then
     SetFillStyle(8,2)
  Else
     SetFillStyle(8,1);
  FloodFill(0,0,1);
  SwapGraph(TRUE);
  Readln;
  RestoreCrtMode;
  If (EmmInstalled) Then
     DisplayEmmInfo
  Else
     Writeln('The image has been swapped to RAM.');
  Readln;
  SetGraphMode(gMode);
  SwapGraph(FALSE);
  Readln;
  CloseGraph;
end.
```

รูปที่ 2 (ต่อ)

ของ EMS และเมื่อกดแป้น <ENTER> อีกครั้ง โปรแกรมจะกลับเข้าสู่กราฟิกโหมด และเรียกข้อมูลจอภาพจาก EMS มาแสดง ในกรณีที่ไม่มี EMS หรือมีจำหนวน Pages ไม่เพียงพอ โปรแกรมจะเก็บข้อมูลจอภาพไว้ใน RAM แทน ผู้เขียนหวังว่าผู้อ่านคงทำความเข้าใจกับโปรแกรมได้ไม่ยาก

การใช้ EMS ไม่เพียงแต่จะใช้เก็บข้อมูลเท่านั้น ยังสามารถนำมาเก็บ Code, Overlay, Resident Program และอื่น ๆ ได้อีก ผู้เขียนหวังว่าโปรแกรมดังกล่าวจะเป็นประโยชน์แก่ผู้อ่านในการนำไปประยุกต์ใช้ในงานของทุกท่าน...สวัสดีครับ